**什么是区域**

由于国家和本地习惯的不同，在处理文字、日期、数字、货币格式等问题都会有所不同，计算机中"区域(Locale)"就是用来解决这个问题的。

**区域名的形式**

language\_territory[.codeset]

zh\_CN.UTF-8

en\_US.UTF-8

特殊的区域(C或者POSIX)会让系统表现得与区域无关，对于C或POSIX环境，任何字符集都是允许的。

**主要的区域分类(category)**

|  |  |
| --- | --- |
| LC\_COLLATE | 字符串排序顺序 |
| LC\_CTYPE | 字符分类（什么是一个字符？它的大小写形式是否等效？） |
| LC\_MESSAGES | 消息使用的语言Language of messages |
| LC\_MONETARY | 货币数量使用的格式 |
| LC\_NUMERIC | 数字的格式 |
| LC\_TIME | 日期和时间的格式 |

**OS的区域设置**

Linux：

通过环境变量(LC\_ALL/LC\_\*/LANG设置)

环境变量优先级

LC\_ALL > LC\_\* > LANG

Locale--查看区域

locale -a--查看所有可用区域

Windows：控制面板-时钟和区域-区域

**数据库固有区域相关属性**

encoding：数据库编码，PostgreSQL支持的字符集：http://www.postgres.cn/docs/14/multibyte.html

collate：字符排序，order by、字符列的索引使用

ctype：字符分类，upper、lower、initcap 大小写不敏感的模式匹配和使用字符分类的正则表达式匹配

可通过 \l 或系统表 pg\_database 查看，数据库创建后不可更改。

**postgresql.conf中区域相关参数**

client\_encoding

lc\_messages

lc\_monetary

lc\_numeric

lc\_time

datestyle

default\_text\_search\_config

可通过set命令修改当前session的参数

set lc\_time='en\_US';

**字符编码**

字符编码是字符在计算机中以二进制值表示的方法。比如'a'的ACSII码是0x61。支持简体中文的字符编码包括: UTF8, GB2312(EUC\_CN), GBK,GB18030

**乱码**

字符编码解码的过程中出现错误就会产生乱码。出现乱码可从以下几个方面找原因：

显示的字体和字符编码

处理过程中的编码转换

数据的源头是否有非法字符

**服务端支持的简体中文字符编码**

UTF8,EUC\_CN

**客户端支持的简体中文字符编码**

UTF8,EUC\_CN,GBK,GB18030

**为什么服务端不支持GBK,GB18030 ？**

PostgreSQL使用的flex按照ASCII编码解析SQL语句，ASCII不兼容GBK和GB18030。

**C/S字符编码自动转换**

C/S间编码不一致时服务端自动进行字符编码转换。

要想启用自动字符集转换功能，必须告诉PostgreSQL在客户端使用的字符集（编码）。

1.用psql里的\encoding命令。

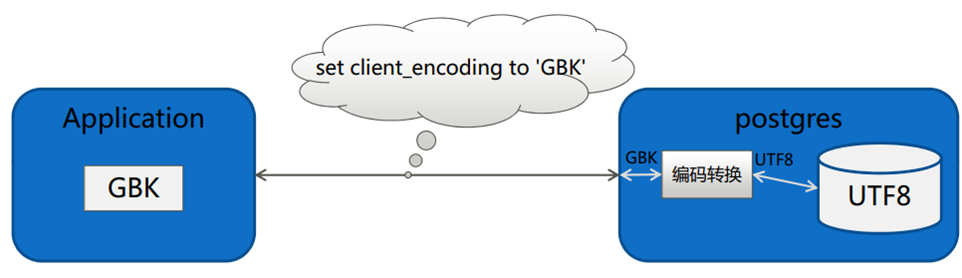
\encoding UTF8

2.使用SET client\_encoding TO设置客户端编码

SET NAMES 'value';--标准SQL语法

RESET client\_encoding;--重置到缺省编码

3. 在客户端的环境里定义了PGCLIENTENCODING环境变量。



**服务端的字符编码检查**

PostgreSQL服务端对不符合数据库编码的字符报错。

试图在EUC\_CN的数据库上使用EUC\_CN不支持的"丂"字时PostgreSQL报错

show server\_encoding;

show client\_encoding;

select '丂';--报错

在UTF8的数据库上可以使用"丂"字

**排序规则**

排序规则在PostgreSQL中的影响

字符比较

ORDER BY

字符列的索引使用

PostgreSQL中字符比较的实现

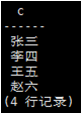
C或POSIX：按照数据库字符编码下的字符二进制代码比较

其他区域：通过调用操作系统提供的strcoll()实现字符比较。中文区域通常按拼音比较。

**例1：排序**

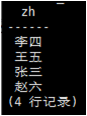
LC\_COLLATE=C (按汉字编码排序)

select \* from (values ('赵六'),('张三'),('李四'),('王五')) as a (c) order by c collate "C";



LC\_COLLATE=zh\_CN

(按汉字拼音排序)



**例2:索引**

查询的collate和索引的collate一致时会使用索引：

explain select \* from t1 where name='a';

查询的collate和索引的collate不一致时不会使用索引：

explain select \* from t1 where name='a' collate "zh\_CN";

**例3：前缀模糊匹配**

show lc\_collate;--C

explain analyze select count(\*) from t1 where col like 'xxx%';-- Index Scan

show lc\_collate;--zh\_CN

explain analyze select count(\*) from t1 where col like 'xxx%';-- Seq Scan

如果collate不为"C"，但期望LIKE 或者 POSIX 正则表达式，使用索引，则需要指定text\_pattern\_ops,varchar\_pattern\_ops 和 bpchar\_pattern\_ops 操作符类。

create index idx\_t1\_name on t1(name text\_pattern\_ops);

**排序规则的设置**

作为数据库属性的排序规则不能变更，但可以在列和表达式上指定不同的排序规则

CREATE TABLE test1 (

a text COLLATE "C",

b text COLLATE "zh\_CN",

...

);

**指定表达式的排序规则**

create index on test1(a collate "zh\_CN");

select \* from test1 order by a collate "zh\_CN";

**Collate的优先级**

表达式 > 列定义 > 数据库定义

**可动态指定的排序规则一览**

可以在列和表达式上指定的"COLLATE" 是定义在pg\_collation中的对象，相当于 LC\_COLLATE+LC\_CTYPE 组合的别名。

select pg\_encoding\_to\_char(collencoding) as encoding,collname,collcollate,collctype from pg\_collation;

返回结果如下所示，encoding为空时，表示这个collation支持所有的字符集。

**中文拼音排序规则的缺陷**

zh\_CN字符比较的性能比C差很多

恢复数据：性能差1倍

vacuum analyze:性能差5倍

建索引:性能差5倍

**Collate对性能的影响**

create table t1(info text);

insert into t1 select md5(generate\_series(1,1000000)::text);

vacuum ANALYZE tb1;

select \* from t1 order by info limit 1;

create index idx1 on tb1(c1);

测试结果

Collate/Ctype C en\_US.UTF8 zh\_CN.UTF8

insert

vacuum ANALYZE

order by

create index

insert(with index)

某些多音字的拼音排序结果不一定符合预期

Linux 下平台的中文排序：

select \* from (values ('阿'),('啊'),('腌'),('包'),('')) as a (e) order by e;

返回结果

""

"腌"

"阿"

"啊"

"包"

这里的 "腌" 是多音字，取 "za" 发音。不是 yan 音。

依赖于操作系统实现，不同OS下结果可能不一样

依赖于操作系统实现不同，字母排序可能会出现：

A a B b C c ... 而不是 A B C ... a b c ...

解决方法实现进行 ascii 字符比较，无法得出结果再进行编码转化比较。

select \* from (values ('A'),('B'),('C'),('a'),('b'),('c')) as a (e) order by e

考虑到性能，可以把数据库的Collate设成C，只在真正需要按拼音排序的列或者表达式上使用Collate "zh\_CN"。

**字符分类(ctype)**

影响的功能

upper(),lower(),initcap()

大小写不敏感的模式匹配

使用了字符分类的正则表达式匹配

例:

LC\_CTYPE为C时,不能识别全角英文字母。

select upper('ａ');--a

select lower('Ａ');--A

select initcap('ａａａ');--ａａａ

select 'ａ' ilike 'Ａ';-- f

LC\_CTYPE为zh\_CN时，可以识别全角英文字母 。

select upper('ａ');--Ａ

select lower('Ａ');--ａ

select initcap('ａａａ');--Ａａａ

select 'ａ' ilike 'Ａ';-- t

然而对正则表达式的字符分类，不论区域是什么都不识别全角英文字母。

[:upper:]：代表大写字符，即 A-Z

[:alnum:]：代表英文大小写字符及数字，即 0-9，A-Z，a-z

select 'Ａ' collate "C" ~ '[[:upper:]]';--f

select '１' collate "C" ~ '[[:alnum:]]';--f

select 'Ａ' collate "C" ~ '[[:upper:]]';--f

select '１' collate "C" ~ '[[:alnum:]]';--f

**但是OS是支持的。**

export LC\_ALL=C

echo Ａ | grep -E [[:upper:]]

echo １ | grep -E [[:alnum:]]

export LC\_ALL=zh\_CN.utf8

echo Ａ | grep -E [[:upper:]] --Ａ

echo １ | grep -E [[:alnum:]] --１

Ctype会影响pg\_trgm，比如Ctype为'C'时，pg\_trgm将无法支持中文

Ctype="C"
select show\_trgm('aaabbbc到的x');
show\_trgm

{" a"," x"," aa"," x ",aaa,aab,abb,bbb,bbc,"bc "}

Ctype="en\_US"
select show\_trgm('aaabbbc到的x');
show\_trgm

{" a"," aa",0x27bdf1,0x30bd19,0x4624bc,aaa,aab,abb,bbb,bbc,0x6a2ad5}

Ctype="zh\_CN"
select show\_trgm('aaabbbc到的x');
show\_trgm

{" a"," aa",0x27bdf1,0x30bd19,0x4624bc,aaa,aab,abb,bbb,bbc,0x6a2ad5}

**消息使用的语言**

**影响的功能**

Log中输出消息的语言

客户端捕获的服务端消息的语言

--enable-nls：支持的自然语言，如果不指定，则安装所有可用的语言包

./configure --prefix=/usr/local/pgsql14.5 --with-perl --with-python --with-libxml --with-libxslt --with-openssl --enable-nls=zh\_CN

set lc\_messages="C";

select xx;--ERROR: column "xx" does not exist

set lc\_messages="zh\_CN";

select xx;--错误: 字段 "xx" 不存在

**输出中文消息时的注意点**

服务端消息日志可能会包含不同字符编码的消息

不同字符编码的多个数据库的输出的消息编码不同。

create database testdb encoding latin1 locale "C" template template0;

客户端得到的连接错误消息可能是乱码

连接时服务端还不知道客户端的client\_encoding，如果客户端和服务库编码不一致可能输出中文消息时的注意点会导致乱码。同理，数据库名，用户名和密码中如果有中文，也可能会导致客户端无法建立连接。

**数值输出格式**

LC\_MONETARY影响money数据类型的输出

set lc\_monetary="C";

select 1::money;-- $1.00

set lc\_monetary="zh\_CN";

select 1::money;-- ￥1.00

(1 行记录)

Money类型比较鸡肋，不如使用numeric

**LC\_NUMERIC**

set lc\_numeric="C";

select 1.1::numeric;-- 1.1

set lc\_numeric="zh\_CN";

select 1.1::numeric;-- 1.1

**日期和时间的格式**

**DateStyle对时间输入的影响**

show DateStyle;

set DateStyle="ISO,YMD";

select '01/02/03'::date;

set DateStyle="ISO,MDY";

select '01/02/03'::date;

**DateStyle对时间输出的影响**

set DateStyle="ISO,MDY";

select now();-- 2022-11-16 17:16:29.730246+08

set datestyle="sql";

select now();-- 11/16/2022 17:16:49.748607 CST

**LC\_TIME似乎对时间没有影响**

set lc\_time="C";

select now();-- 2022-11-16 17:12:40.833641+08

set lc\_time="zh\_CN";

select now();-- 2022-11-16 17:12:40.833641+08

**总结**

**相关建议**

服务端使用UTF8作为数据库编码

客户端设置合适的区域环境和client\_encoding

数据库的Collate设成C，然后只在真正需要按拼音排序的列或者表达式上使用Collate"zh\_CN"。

使用标准的区域无关的数据输出格式

创建数据库集群：

initdb -E UTF8 --locale=C data\_dir

启动数据库：

export LC\_ALL=zh\_CN.UTF8

pg\_ctl -D data -l logfile start

客户端建立连接：

export LC\_ALL=zh\_CN.UTF8

psql

**设置数据库的本地化（collate）示例**

**设置字段的本地化**

1.建表时指定

CREATE TABLE t1 (

id int,

name text COLLATE “zh\_CN",

age int,

);

2.修改列collate

注意：修改列collate时，会导致rewrite table，大表请谨慎操作。

alter table t1 alter name type text COLLATE “C";

**在SQL使用本地化**

使用本地化，改变order by输出排序。命令如下：

select \* from t1 order by name collate "zh\_CN";

select \* from t1 order by name collate "en\_US";--根据en\_US规则进行排序

**使用本地化，改变操作符的结果**

select \* from a where t1 > 'xxx' collate "zh\_CN";

select \* from a where t1 > 'xxx' collate "en\_US";

**使用本土化索引进行排序**

排序语句中的collate与索引的collate保持一致，才能使用这个索引进行排序

create index idx\_t1\_name on t1(name collate "zh\_CN");

explain select \* from t1 order by name collate "zh\_CN";

**设置输出结果按拼音排序**

可以通过如下四种方法来设置按拼音排序：

1.使用本地化 SQL。该方法不修改原有数据。

select \* from t1 order by name collate "en\_US";

2.使用本土化字段。若已有数据，使用该方法时需要调整原有数据。

alter table t1 alter t1 type text COLLATE "en\_US";

3.使用本土化索引以及本土化 SQL。该方法不修改原有数据。

create index idx\_t1\_name on t1(name collate "en\_US");

explain select \* from t1 order by name collate "en\_US";

4.将数据库的collate设置为en\_US，数据会将默认使用这个collate按拼音排序。

create database testdb encoding 'UTF8' lc\_collate 'en\_US.utf8' lc\_ctype 'en\_US.utf8' template template0;

\c testdb

select \* from (values ('Alice'),('Tom')) as a(c) order by c ;

注意：如果是中文，在设置按拼音排序时，要注意多音字。例如重庆（chongqing），在编码时，重可能会按照zhong编码，影响输出。